MeTAGeM

A Meta-Tool for the Automatic Generation

of Model Transformation

User Guide

- Version 1.0 -

October 2010

Content

List of Figures

1. Introduction

MeTAGeM is a Meta-Tool for the Automatic Generation of Model Transformation that makes possible to put into practice the MDE (Model-Driven Engineering) principles for developing model transformations, defining model transformations as models, without considering details of the code implementation.

In order to do that, MeTAGeM gives you a DSLs (Domain-Specific Languages) collection that make it possible to model the high-level model transformations as well as the transformations between models of different levels and its subsequent code generation.

The transformation modelling is divided in four abstraction levels:

- **Platform-independent level,** where the user must define the relations between elements of the meta-models implicated in the transformation.
- **Platform-specific level,** which is defined following a hybrid approach.
- **Platform-definition level,** where the user can choose a specific transformation language. In this version of MeTAGeM Tool, the user can choose between the ATL or RubyTL transformation languages.
- **Code level** implements the transformation into the chosen transformation language.

2. Installation and System Requirements

To download and install MeTAGeM in your computer please follow the steps below:

1. Install Java Virtual Machine (1.6+)

Available in: http://www.java.com/es/download/index.jsp

2. Download and Install Eclipse and MeTAGeM

In this case, there are two possibilities:

1) **Recommended**: Download Eclipse AMMA-MeTAGeM. This distribution contains all plug-ins required to work correctly with MeTAGeM. It is available in:
http://www.kybele.etsii.urjc.es/members/vbollati/thesis/Software/EclipseAMMA_MeTAGeM.zip

2) Download Eclipse Classic (http://www.eclipse.org/downloads) and then install the following plug-ins:
   - MeTAGeM 1.0
     (http://www.kybele.etsii.urjc.es/members/vbollati/thesis/Software/MeTAGeM-Plugins.zip)
   - AGE 0.3.4
   - AM3 0.4
   - AMW 1.0
   - ATL 2.0
   - EMF 2.4
   - Epsilon 0.8.6
   - RubyTL 0.3.4
   - TCS 0.8

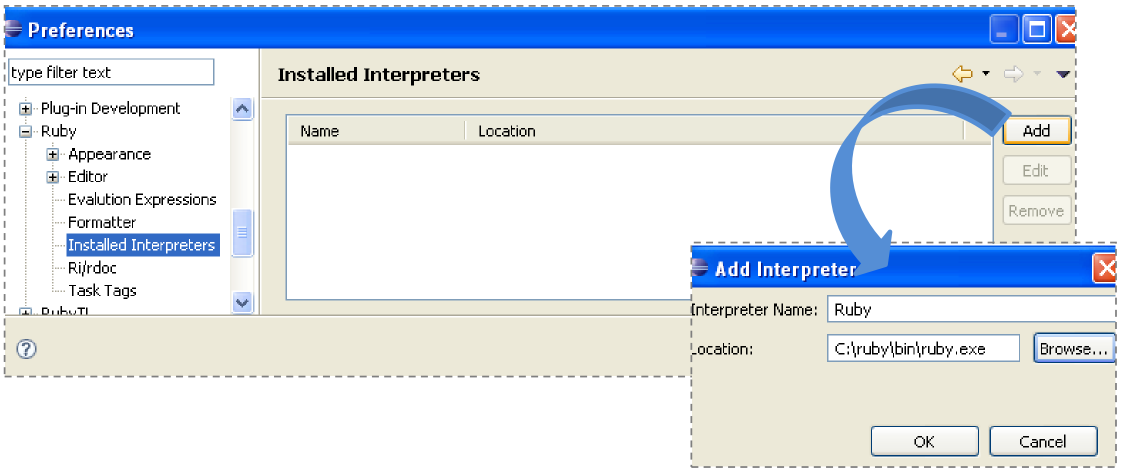
3. Install Ruby Interpreter (1.8.6 version)

To run RubyTL, Ruby Runtime must be installed and it is recommended the 1.8.6 version.

If you have downloaded Eclipse AMMA-MeTAGeM, you can find the installer (*ruby186-25.exe*) in the *RubyTL* folder. In other case, you can download it from:

http://rubyforge.org/frs/download.php/18566/ruby186-25.exe

Once the interpreter is installed, AGE should be configured with the path where the Ruby interpreter has been installed. Click on Windows → Preferences → Ruby → Installed Interpreters preferences. As shown Figure 1, in this dialog, you can add the installed interpreter, usually the path c:/ruby/bin/ruby.exe or /usr/bin/ruby is used.



Interpreter Name: Ruby
Location: C:\ruby\bin\ruby.exe Browse...

Fig. 1 – Ruby Interpreter configuration

3. Creating a MeTAGeM Model

For modelling platform-independent transformations, MeTAGeM Tool provides an editor of models that makes it possible to represent the relationships between elements of the meta-models that are implicated in the transformation.

To start with transformation modelling in this level, you must define a model conforms to the extended *weaving* meta-model of MeTAGeM. To create such model, you must click on *File→New→Other→Model Weaver→Weaving Model*, as Figure 2 shows.

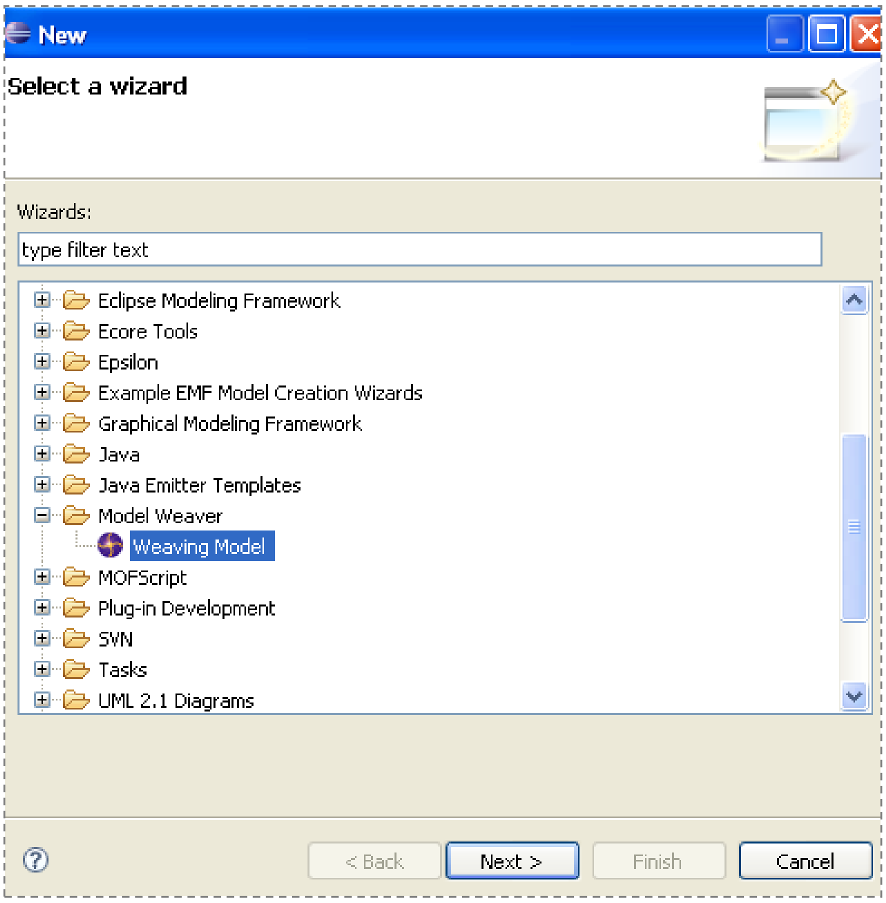
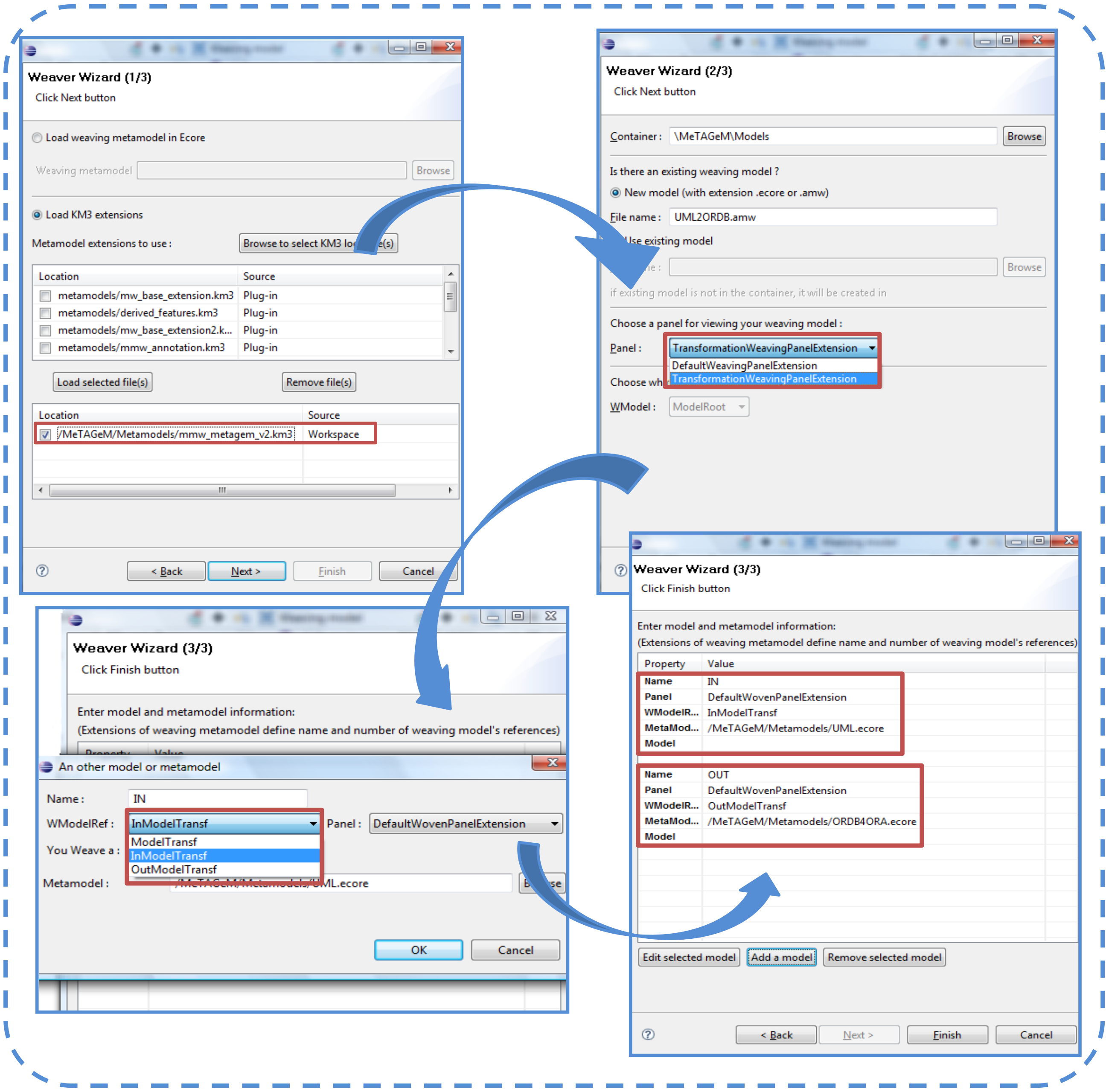


Fig. 2 – Selecting a *weaving* model

After selecting 'Weaving Model', the wizard provides you all the steps to create a new weaving model; in the Figure 3 you can see all this steps.

First, you must select the plug-in that you will use to define the model, so in this case, you must select MeTAGeM plug-in (*Metamodels/mmw_metagem_v2.km3*). The following step corresponds to indicate where you want to save the new model (*Container*), its name (*FileName*), and the kind of panel that in this case must be *TransformationWeavingPanelExtension.*

The last step is selecting the meta-models that you want to transform: the source meta-model and the target meta-model; as well as the name of these meta-models in the transformation scope.



Weaver Wizard (1/3)
Click Next button

Load weaving metamodel in Ecore
Weaving metamodel
Load KM3 extensions
Metamodel extensions to use :

Location	Source
metamodels/mw_base_extension.km3	Plug-in
metamodels/derived_features.km3	Plug-in
metamodels/mw_base_extension2.k...	Plug-in
metamodels/mmw_annotation.km3	Plug-in

Location	Source
✓ /MeTAGeM/Metamodels/mmw_metagem_v2.km3	Workspace

Weaver Wizard (2/3)
Click Next button

Container : \MeTAGeM\Models
Is there an existing weaving model ?
New model (with extension .ecore or .amw)
File name : UML2ORDB.amw
Use existing model
if existing model is not in the container, it will be created in
Choose a panel for viewing your weaving model :
Panel : TransformationWeavingPanelExtension
DefaultWeavingPanelExtension
TransformationWeavingPanelExtension
WModel : ModelRoot

Weaver Wizard (3/3)
Click Finish button

Enter model and metamodel information:
(Extensions of weaving metamodel define name and number of weaving model's references)

An other model or metamodel
Name : IN
WModelRef : InModelTransf
ModelTransf
InModelTransf
OutModelTransf
Panel : DefaultWovenPanelExtension
You Weave a :
Metamodel : /MeTAGeM/Metamodels/UML.ecore

Weaver Wizard (3/3)
Click Finish button

Enter model and metamodel information:
(Extensions of weaving metamodel define name and number of weaving model's references)

Property	Value
Name	IN
Panel	DefaultWovenPanelExtension
WModelR...	InModelTransf
MetaMod...	/MeTAGeM/Metamodels/UML.ecore
Model	
Name	OUT
Panel	DefaultWovenPanelExtension
WModelR...	OutModelTransf
MetaMod...	/MeTAGeM/Metamodels/ORDB4ORA.ecore
Model	

Fig. 3 – Creating a MeTAGeM weaving model

After finishing this process, you will obtain the new model where you can define relationship between elements of the meta-models selected. In order to do this, MeTAGeM Tool provides a *weaving editor* that simplifies this task. As you can see in the Figure 4, the left panel shows the source meta-model (in this example: SQL2003 meta-model); the right panel shows the target meta-model (in this example:

ORDB4ORA meta-model); and the center panel shows the relationships between elements of these meta-models.

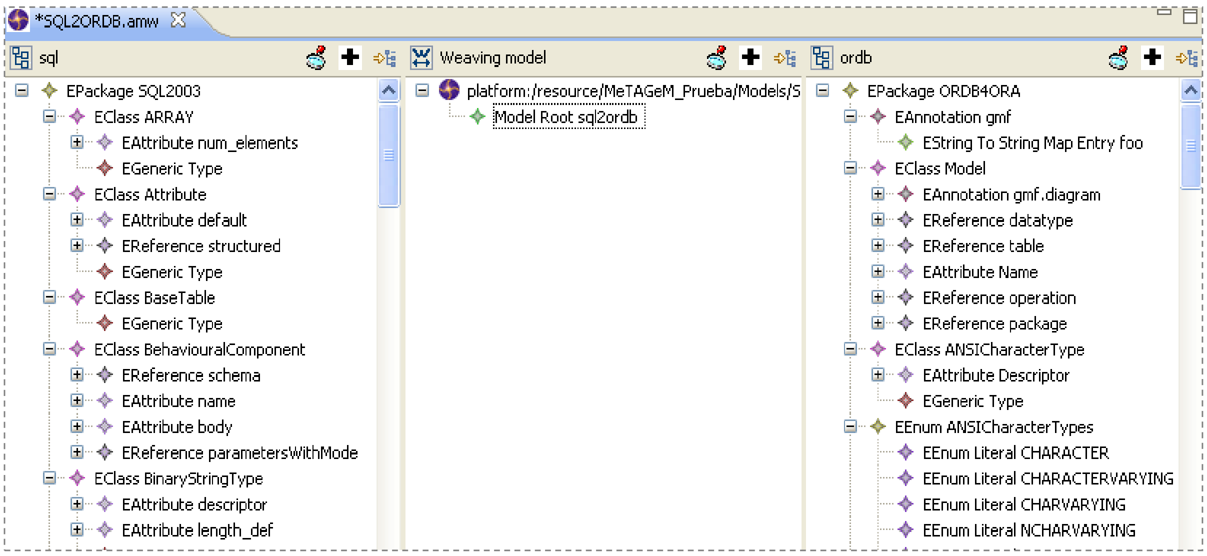


Fig. 4 – MeTAGeM weaving model

Now, you can define the relationships clicking on the secondary mouse button on the model root and then, select a new child.

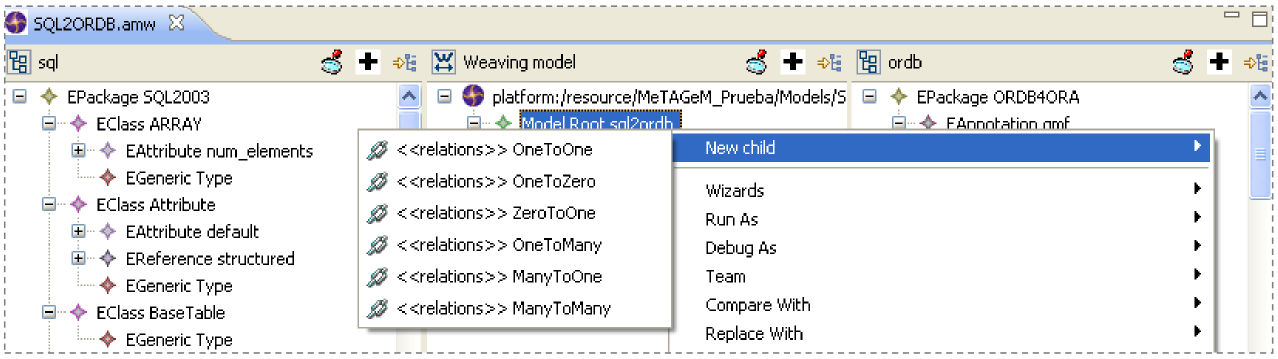


Fig. 5 – Creating a new relationship in a MeTAGeM model

MeTAGeM gives you the possibility to create the following kind of relationships:

- **OneToOne:** Defines a relationship between one source element and one target element.
- **OneToZero:** Defines a relationship with only one source element.
- **ZeroToOne:** Defines a relationship with only one target element.
- **OneToMany:** Defines a relationship between one source element and several target elements.
- **ManyToOne:** Defines a relationship between several source elements and only one target element.
- **ManyToMany:** Defines a relationship between several source elements and several target elements.

After defining a relationship, you must select its source and its target elements, in order to do it, you must drag an element from left or right panel and drop it over the relationship, as Figure 6 shows.

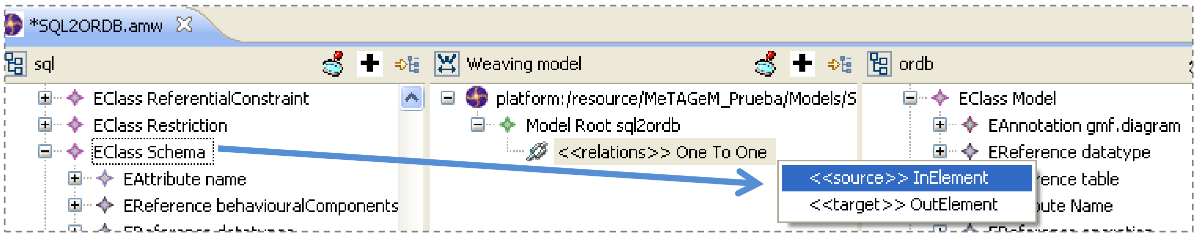


Fig. 6 – Selecting a source element for a relationship

Also, as you can see in Figure 7, an *OutElement* can have internal relationships. This feature makes MeTAGeM capable of representing relationships between properties of the relation elements. In this context, you can only define the following relationships: *OneToOne, ManyToOne* and *ZeroToOne*.

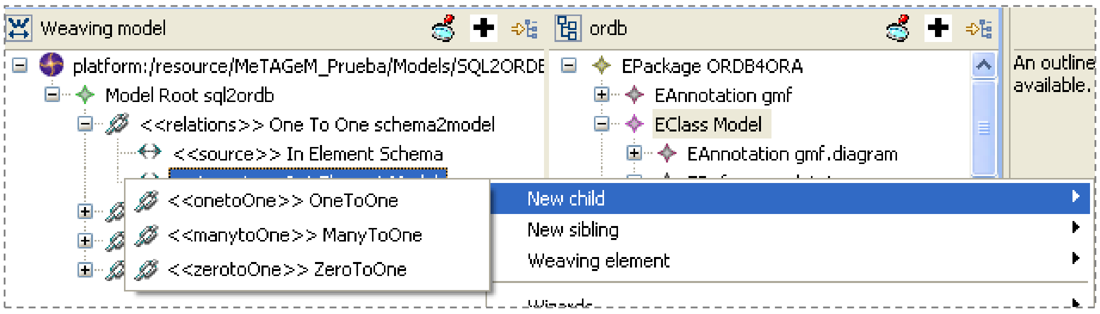
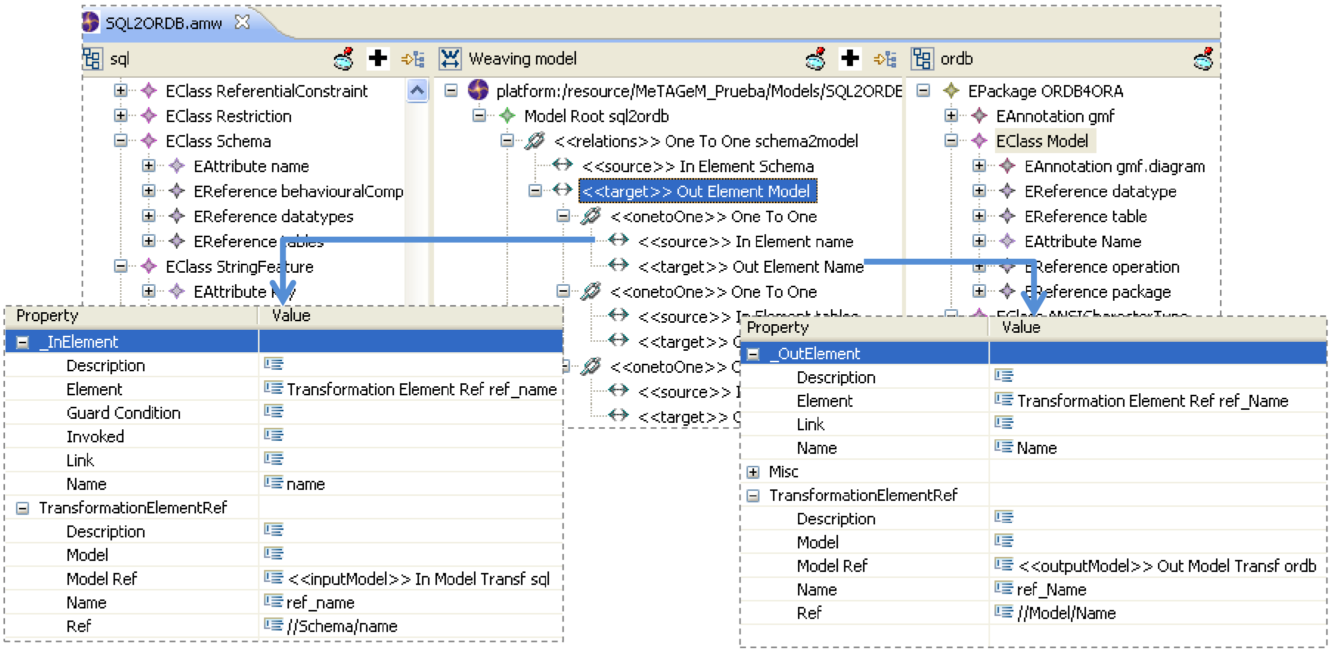


Fig. 7 – Relationships included in another relationship (MeTAGeM model)

Now, you are able to define a MeTAGeM model like this:



Property	Value
_InElement	
Description	
Element	Transformation Element Ref ref_name
Guard Condition	
Invoked	
Link	
Name	name
TransformationElementRef	
Description	
Model	
Model Ref	<<inputModel>> In Model Transf sql
Name	ref_name
Ref	//Schema/name

Property	Value
_OutElement	
Description	
Element	Transformation Element Ref ref_Name
Link	
Name	Name
Misc	
TransformationElementRef	
Description	
Model	
Model Ref	<<outputModel>> Out Model Transf ordb
Name	ref_Name
Ref	//Model/Name

Fig. 8 – MeTAGeM model example

In the above figure, we have defined an *OneToOne* relationship (schema2model) that contains a source element (Schema) and a target element (Model). This target element also contains three relationships that represent different properties of the relationship between *Schema* and *Model*. In this way, for example, we have specified that *Name* property of *Schema* element will be transformed to *Name* property of *Model* element.

4. Generating an Hybrid Model

After creating the MeTAGeM Model (AMW [1]), you can generate the Hybrid Model. To do this, you must launch the *MeTAGeM->Hybrid* process (Run As → MeTAGeM ->Hybrid) as Figure 9 shows.

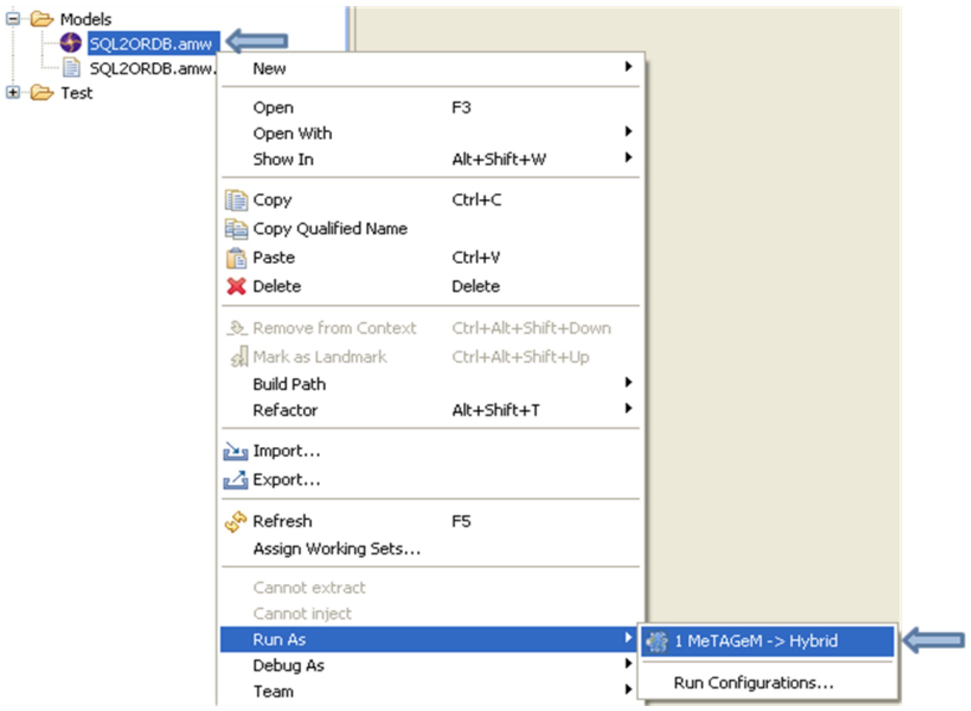
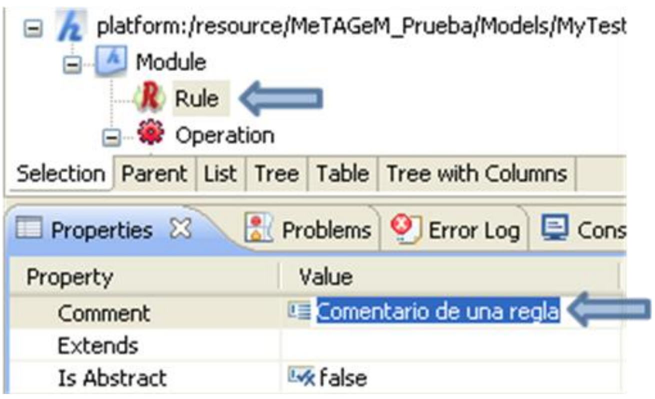


Fig. 9 – Generating Hybrid Model from MeTAGeM Model

Now, you have a Hybrid Model which represents the transformation in a lower abstraction level. Due to its lower-level, Hybrid meta-model allows you to add the following new features to the transformation:

- **Add Comments in the rules**



Property	Value
Comment	Comentario de una regla
Extends	
Is Abstract	false

Fig. 10 – Comment in a rule (Hybrid Model)

- **Create Operations**

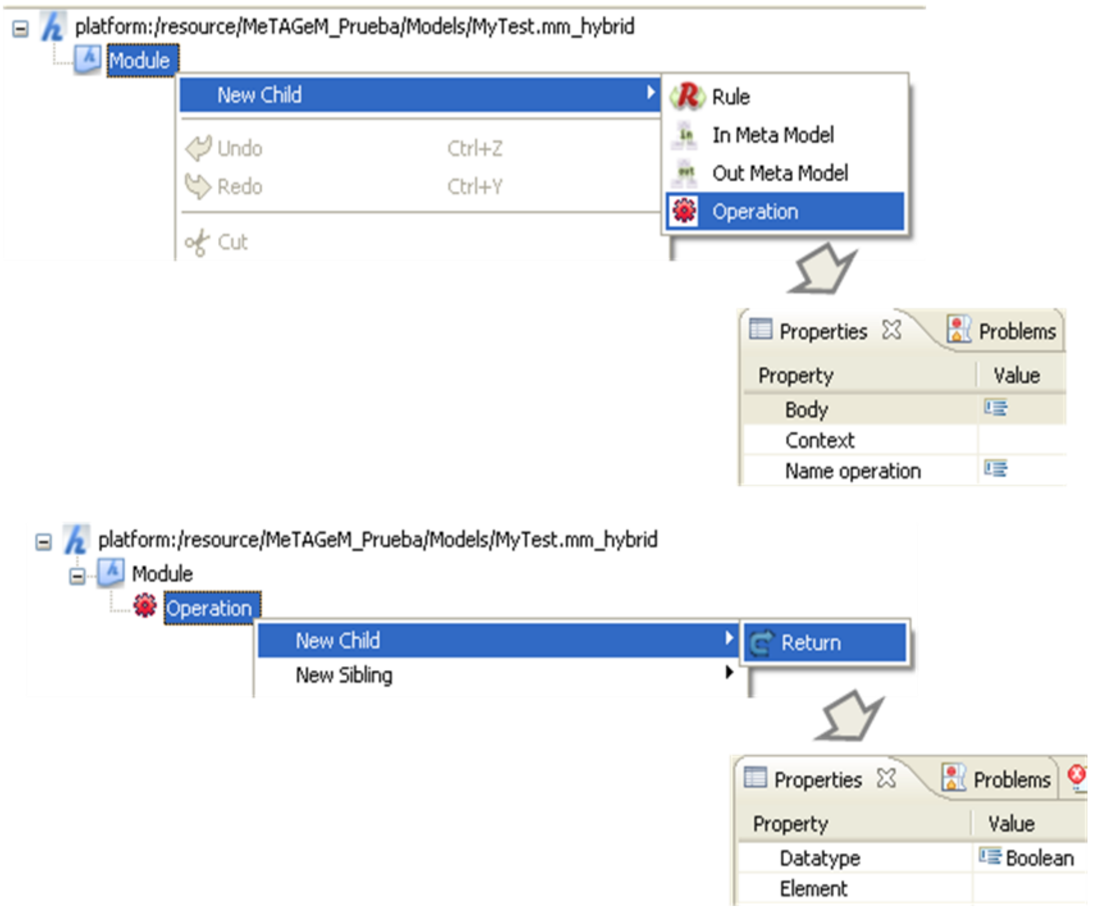
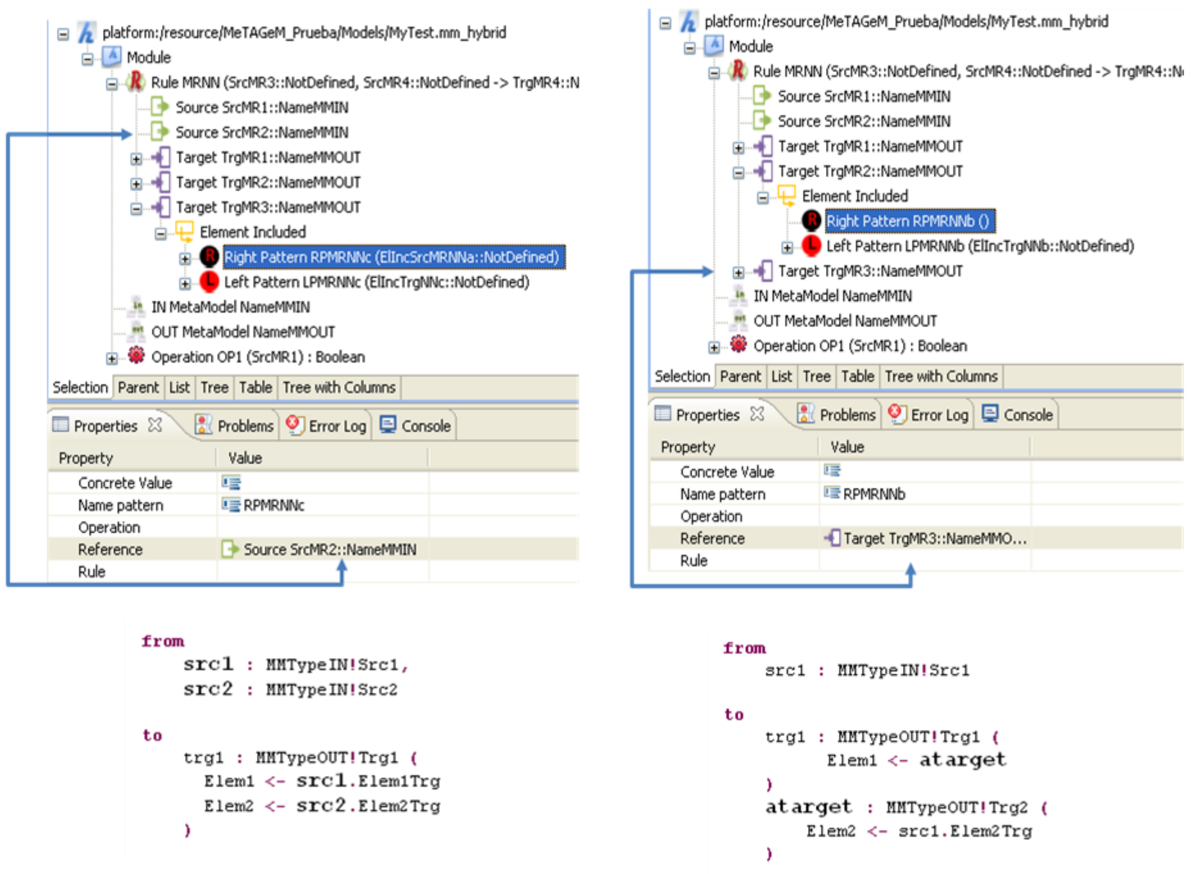


Fig. 11 – Creating an operation (Hybrid Model)

- **Relation "reference-isRefered".** This relation is defined between *RightPattern* and *Element* metaclasses and it performs two functions:

3) When the *Element* is a *SourceElementRule*, this relation allows you to associate a *Binding* with one source element of the rule. This role is recommended when a rule has many sources.
4) When the *Element* is a *TargetElementRule,* this relation indicates that the source of the *Binding* is another *Binding* defined in the same rule.

In the Figure 12, you can see an example about Relation "*reference-isRefered*" behaviours and its corresponding ATL code.



```
platform:/resource/MeTAGeM_Prueba/Models/MyTest.mm_hybrid
  Module
    Rule MRNN (SrcMR3::NotDefined, SrcMR4::NotDefined -> TrgMR4::N
      Source SrcMR1::NameMMIN
      Source SrcMR2::NameMMIN
      Target TrgMR1::NameMMOUT
      Target TrgMR2::NameMMOUT
      Target TrgMR3::NameMMOUT
        Element Included
          Right Pattern RPMRNNc (ElIncSrcMRNNa::NotDefined)
          Left Pattern LPMRNNc (ElIncTrgNNc::NotDefined)
    IN MetaModel NameMMIN
    OUT MetaModel NameMMOUT
    Operation OP1 (SrcMR1) : Boolean
```

Property	Value
Concrete Value	
Name pattern	RPMRNNc
Operation	
Reference	Source SrcMR2::NameMMIN
Rule	

```
from
    src1 : MMTypeIN!Src1,
    src2 : MMTypeIN!Src2

to
    trg1 : MMTypeOUT!Trg1 (
      Elem1 <- src1.Elem1Trg
      Elem2 <- src2.Elem2Trg
    )
```

```
platform:/resource/MeTAGeM_Prueba/Models/MyTest.mm_hybrid
  Module
    Rule MRNN (SrcMR3::NotDefined, SrcMR4::NotDefined -> TrgMR4::N
      Source SrcMR1::NameMMIN
      Source SrcMR2::NameMMIN
      Target TrgMR1::NameMMOUT
      Target TrgMR2::NameMMOUT
        Element Included
          Right Pattern RPMRNNb ()
          Left Pattern LPMRNNb (ElIncTrgNNb::NotDefined)
      Target TrgMR3::NameMMOUT
    IN MetaModel NameMMIN
    OUT MetaModel NameMMOUT
    Operation OP1 (SrcMR1) : Boolean
```

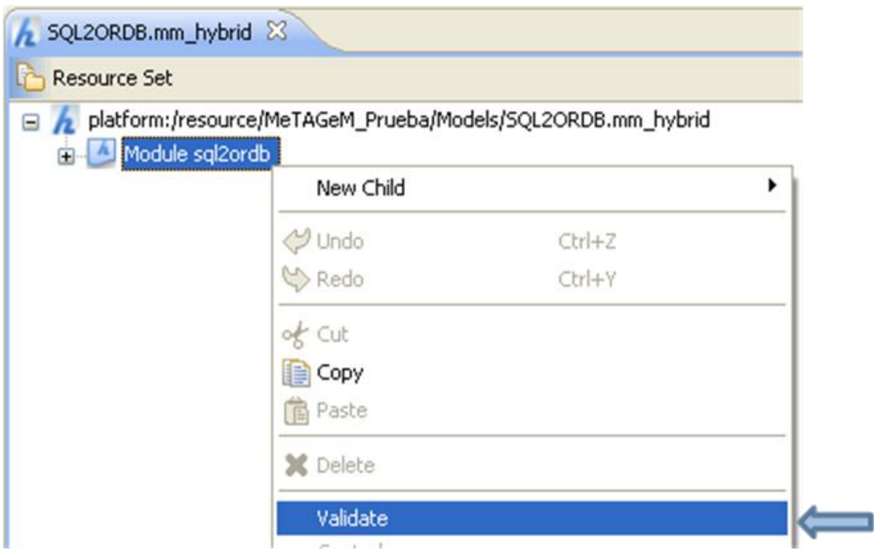
Property	Value
Concrete Value	
Name pattern	RPMRNNb
Operation	
Reference	Target TrgMR3::NameMMO...
Rule	

```
from
    src1 : MMTypeIN!Src1

to
    trg1 : MMTypeOUT!Trg1 (
          Elem1 <- atarget
    )
    atarget : MMTypeOUT!Trg2 (
          Elem2 <- src1.Elem2Trg
    )
```

Fig. 12 – Relation "*reference-isRefered*" roles.

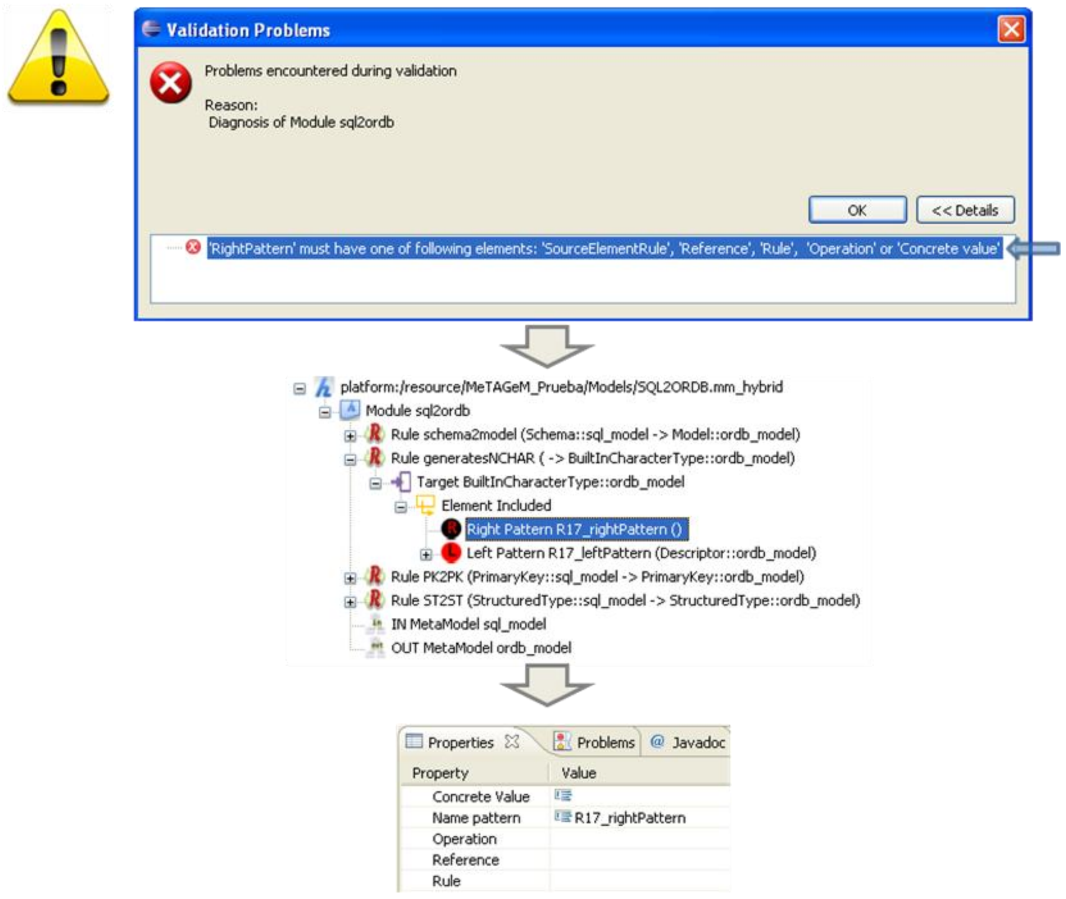
After defining the Hybrid model you must validate it to check there is no error. In order to do it, you must use the contextual menu of the model's root and choose *validate* as you can see in Figure 13.



```
SQL2ORDB.mm_hybrid
Resource Set
platform:/resource/MeTAGeM_Prueba/Models/SQL2ORDB.mm_hybrid
  Module sql2ordb
```

Fig. 13 – Validating Hybrid Model

In case there is an error, the validation process shows an error message as shown in Figure 14. Through this message, you can know what elements or attributes of your model contain validation errors and you can take action to resolve them.


Validation Problems

Problems encountered during validation

Reason:
Diagnosis of Module sql2ordb

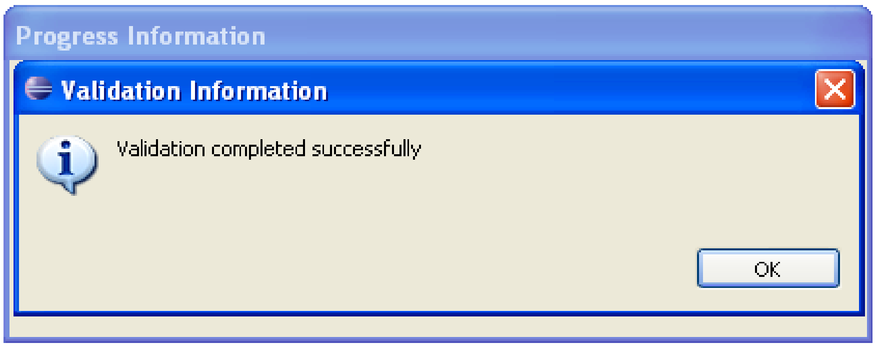
'RightPattern' must have one of following elements: 'SourceElementRule', 'Reference', 'Rule', 'Operation' or 'Concrete value'

```
platform:/resource/MeTAGeM_Prueba/Models/SQL2ORDB.mm_hybrid
  Module sql2ordb
    Rule schema2model (Schema::sql_model -> Model::ordb_model)
    Rule generatesNCHAR ( -> BuiltInCharacterType::ordb_model)
      Target BuiltInCharacterType::ordb_model
        Element Included
          Right Pattern R17_rightPattern ()
          Left Pattern R17_leftPattern (Descriptor::ordb_model)
    Rule PK2PK (PrimaryKey::sql_model -> PrimaryKey::ordb_model)
    Rule ST2ST (StructuredType::sql_model -> StructuredType::ordb_model)
    IN MetaModel sql_model
    OUT MetaModel ordb_model
```

Property	Value
Concrete Value	
Name pattern	R17_rightPattern
Operation	
Reference	
Rule	

Fig. 14 – Error validation example in Hybrid Model

Otherwise, if the model is correct, the validation process shows the following message:


Progress Information

Validation Information

Validation completed successfully

Fig. 15 – Correct validation example

When the Hybrid model is correct, you can already transform it to a specific transformation language model, concretely, MeTAGeM Tool makes you possible to transform Hybrid Model to ATL or to RubyTL, as you can see in the following sections.

5. Hybrid -> ATL

To transform a Hybrid model to ATL model **¡Error! No se encuentra el origen de la referencia.** you must launch the *Hybrid -> ATL* process (Run As → Hybrid -> ATL) as Figure 16 shows:

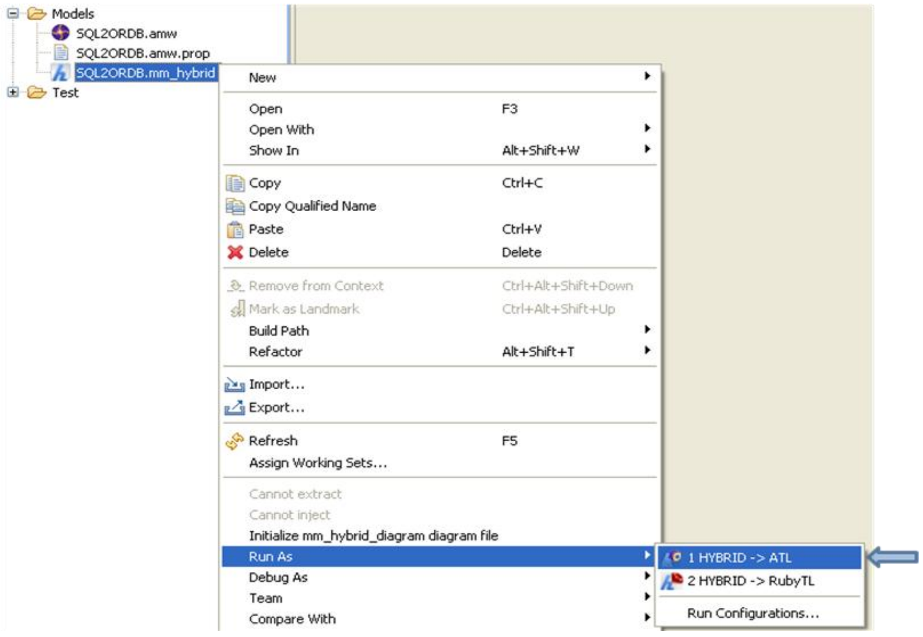
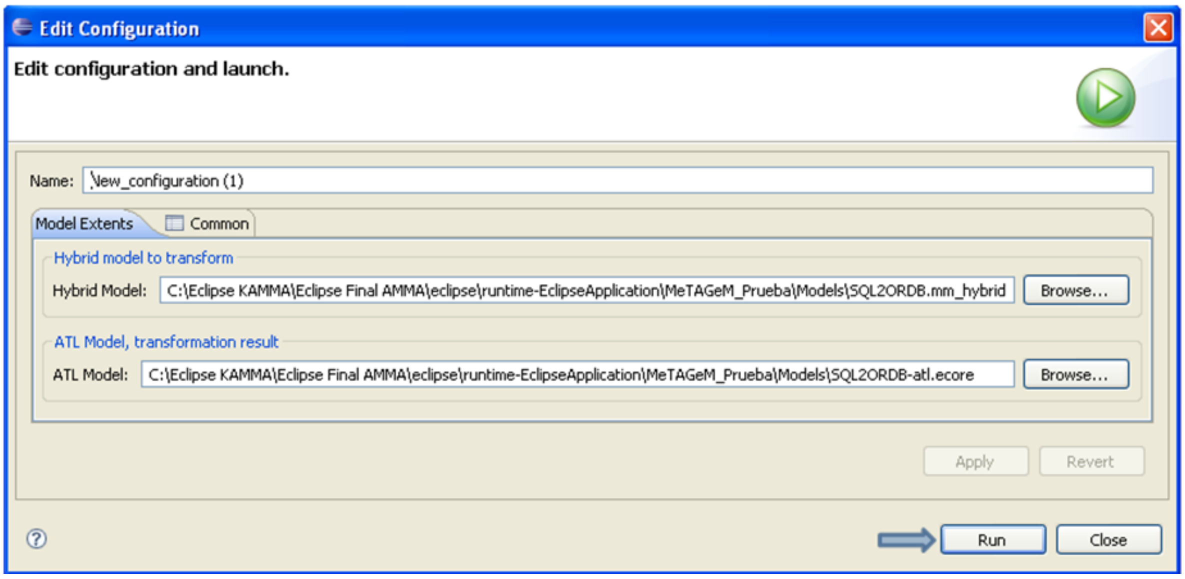


Fig. 16 – Launching *Hybrid -> ATL* Transformation

If it is the first time you launch the process on a Hybrid model you must configure the transformation by choosing the source and target models paths:



Name: New_configuration (1)

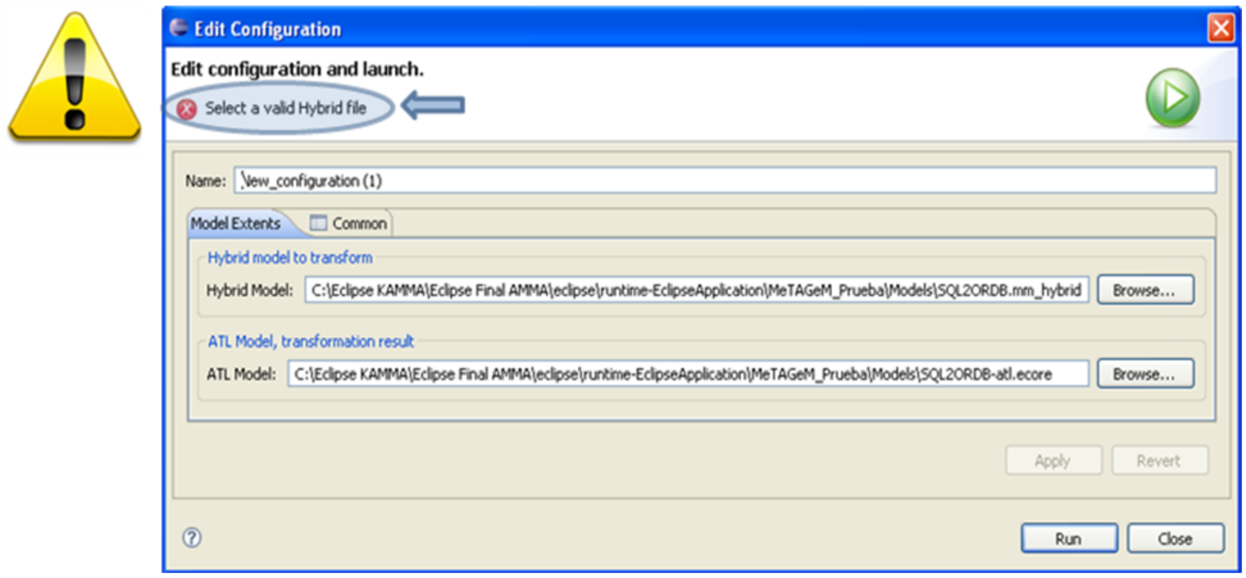
Hybrid model to transform
Hybrid Model: C:\Eclipse KAMMA\Eclipse Final AMMA\eclipse\runtime-EclipseApplication\MeTAGeM_Prueba\Models\SQL2ORDB.mm_hybrid

ATL Model, transformation result
ATL Model: C:\Eclipse KAMMA\Eclipse Final AMMA\eclipse\runtime-EclipseApplication\MeTAGeM_Prueba\Models\SQL2ORDB-atl.ecore

Fig. 17 – Configuring *Hybrid -> ATL* Transformation

The following launches, MeTAGeM will use the same configuration that you have defined in the first launch.

Before running the transformation, the process validates automatically the source model according to the ATL version implemented in MeTAGeM Tool, so it is possible that you have created a valid Hybrid model but that model cannot be transformed to an ATL Model. If it is the first time that you launch the process on a Hybrid model and your hybrid model is not correct according the transformation, you will receive an error message in the *Edit Configuration* window as you can see in the Figure 18.


Edit Configuration

Edit configuration and launch.

Select a valid Hybrid file

Name: New_configuration (1)

Model Extents | Common

Hybrid model to transform

Hybrid Model: C:\Eclipse KAMMA\Eclipse Final AMMA\eclipse\runtime-EclipseApplication\MeTAGeM_Prueba\Models\SQL2ORDB.mm_hybrid

ATL Model, transformation result

ATL Model: C:\Eclipse KAMMA\Eclipse Final AMMA\eclipse\runtime-EclipseApplication\MeTAGeM_Prueba\Models\SQL2ORDB-atl.ecore

Fig. 18 – Validation error in the *Edit Configuration* window (Hybrid->ATL)

In the other hand, if you launch the process from a configuration defined previously and your Hybrid Model is not correct to transform it to ATL, you will receive an error message in the same way that you receive when the Hybrid model is not valid (See Figure 14).

If the transformation is executed correctly, a new ATL model is created in the path chosen previously and then, you can generate the transformation code from this ATL model. To obtain this ATL code, you must click on *Extract ATL-0.2 model to ATL-0.2 file* that appears in the contextual menu of the ATL Model (**Important:** *Extract ATL-0.2 model to ATL-0.2 file* option only appears in AM3 Perspective) as you can see in the Figure 19.

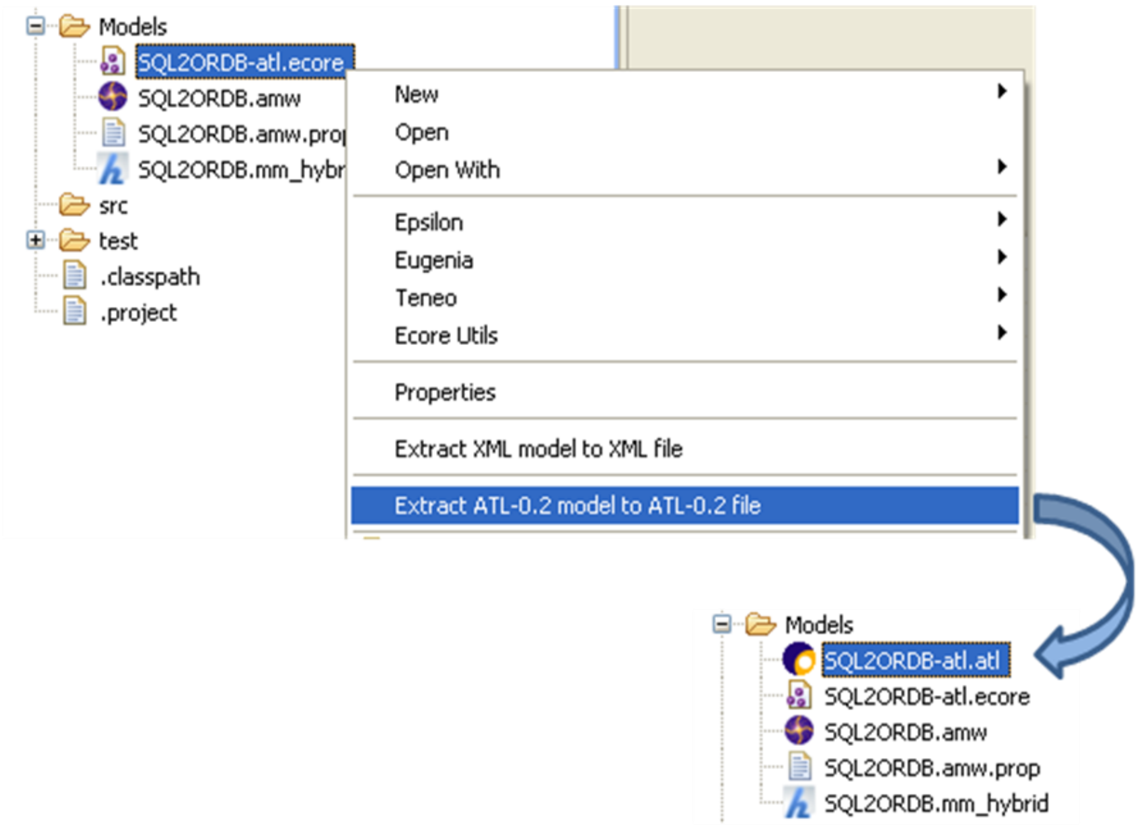


Fig. 19 – Generating ATL code from ATL Model

Now, you have the ATL transformation, but we must notice that this generated code is not always executable, so it is possible that you may need to modify it to achieve a completely executable code. The percentage of the executable code that MeTAGeM Tool generates depends on the complexity of the problem.

6. Hybrid -> RubyTL

From Hybrid Model you can generate an ATL Model and also you can generate a model for another transformation language: RubyTL **¡Error! No se encuentra el origen de la referencia.**. To generate a Ruby Model, you must launch the *Hybrid->RubyTL* process (Run As → Hybrid -> RubyTL) as Figure 20 shows:

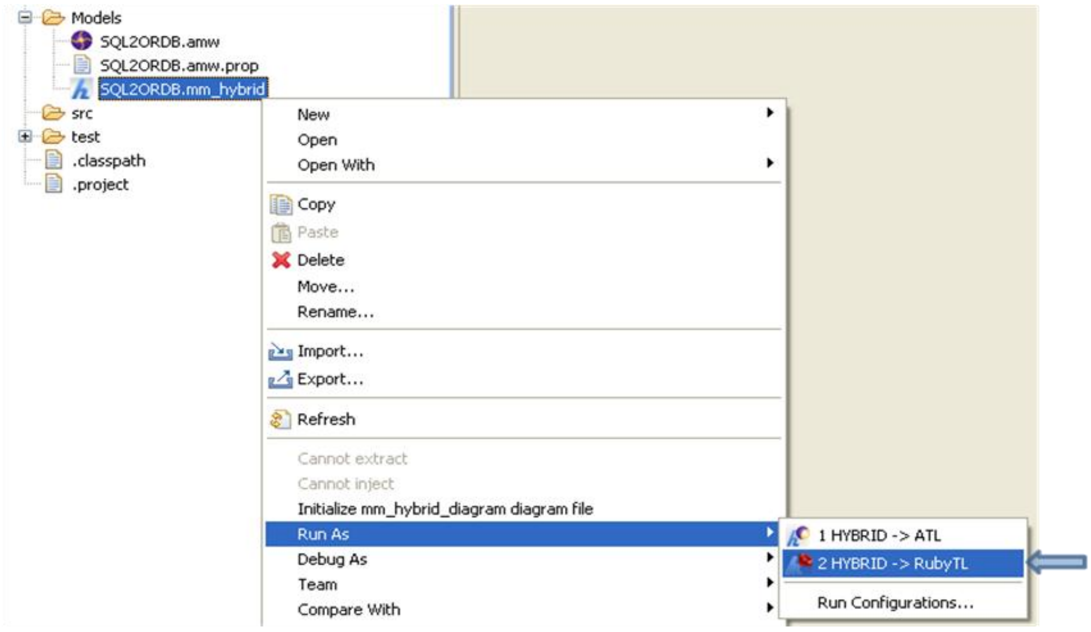
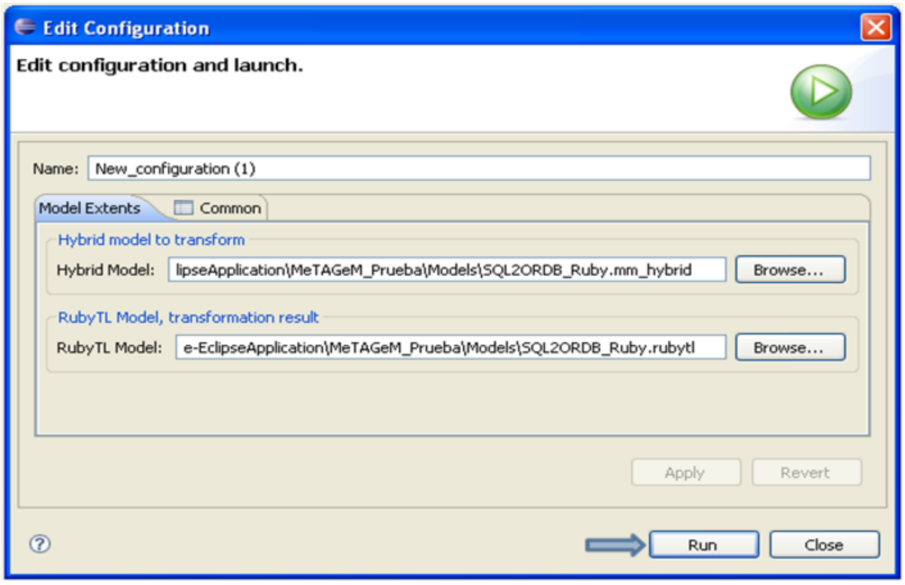


Fig. 20 – Launching *Hybrid -> RubyTL* Transformation

As in the *Hybrid->ATL* case, if it is the first time you launch *Hybrid->RubyTL* process on a Hybrid model, you must configure the transformation by choosing the source and target models paths (see Figure 21).


Edit Configuration

Edit configuration and launch.

Name: New_configuration (1)

Hybrid model to transform

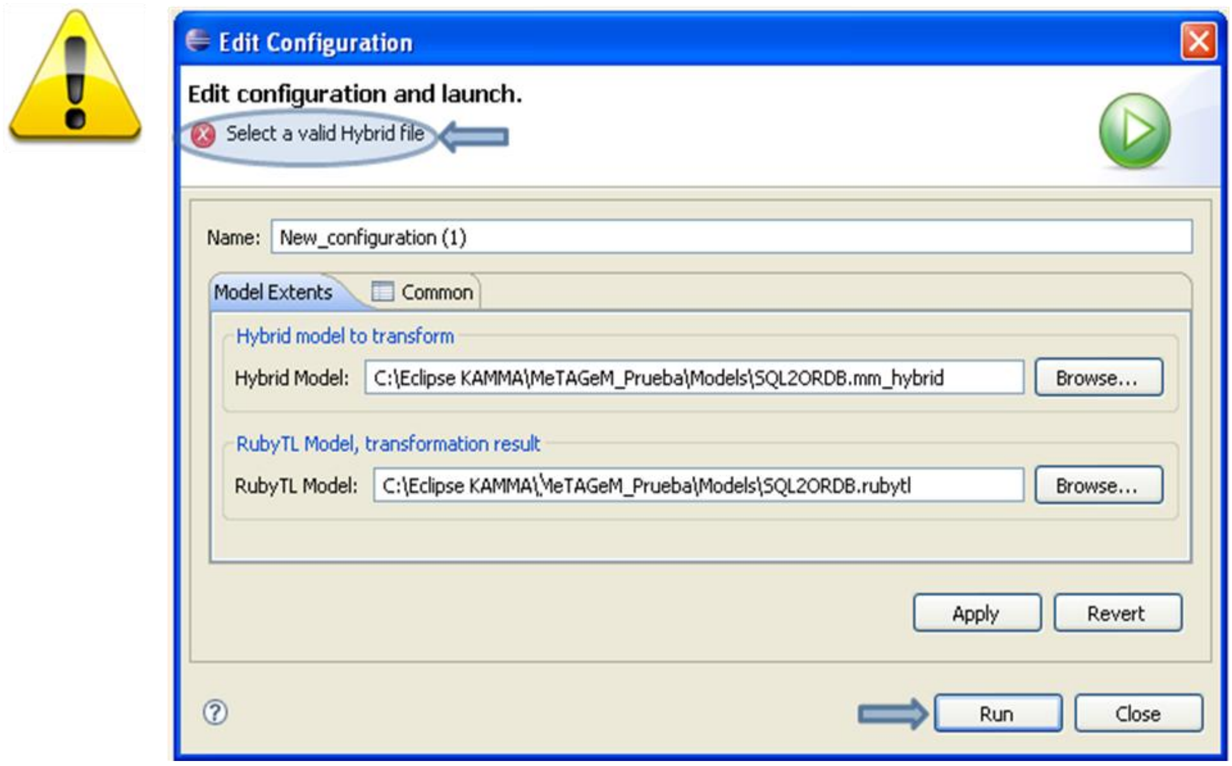
Hybrid Model: lipseApplication\MeTAGeM_Prueba\Models\SQL2ORDB_Ruby.mm_hybrid

RubyTL Model, transformation result

RubyTL Model: e-EclipseApplication\MeTAGeM_Prueba\Models\SQL2ORDB_Ruby.rubytl

Fig. 21 – Configuring *Hybrid -> RubyTL* Transformation

In this case, before running the *Hybrid->RubyTL* process, the MeTAGeM Tool also validates the source model (*Hybrid model*) according to the RubyTL version implemented. If you are configuring the process and your Hybrid model is not valid, you will receive an error message in the *Edit Configuration* window, as Figure 22 shows:


Edit Configuration

Edit configuration and launch.

Select a valid Hybrid file

Name: New_configuration (1)

Hybrid model to transform

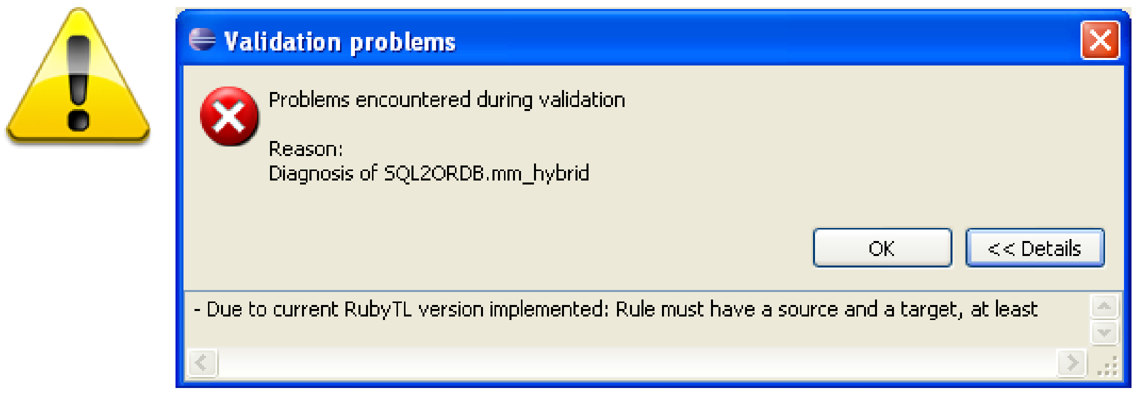
Hybrid Model: C:\Eclipse KAMMA\MeTAGeM_Prueba\Models\SQL2ORDB.mm_hybrid

RubyTL Model, transformation result

RubyTL Model: C:\Eclipse KAMMA\MeTAGeM_Prueba\Models\SQL2ORDB.rubytl

Fig. 22 – Validation error in the *Edit Configuration* window (Hybrid->RubyTL)

However, if you have already configured the process and you launch the transformation (Run As → Hybrid->RubyTL, see Figure 20) and the Hybrid model cannot be transformed to RubyTL, MeTAGeM Tool shows you a validation error, as you can see in the Figure 23, which indicates what the problems of Hybrid model have been found during the validation.


Validation problems

Problems encountered during validation

Reason:
Diagnosis of SQL2ORDB.mm_hybrid

- Due to current RubyTL version implemented: Rule must have a source and a target, at least

Fig. 23 – Validation error example in Hybrid -> RubyTL Transformation

After validating the Hybrid model and if the process runs correctly you will obtain a new model (*rubytl*) which describes the transformation in RubyTL terms. From this model, you can generate the RubyTL transformation code. In order to do it, you must click on *Extract RubyTL model to RubyTL file (MeTAGeM)* that appears in the contextual menu of RubyTL model, as Figure 24 shows:

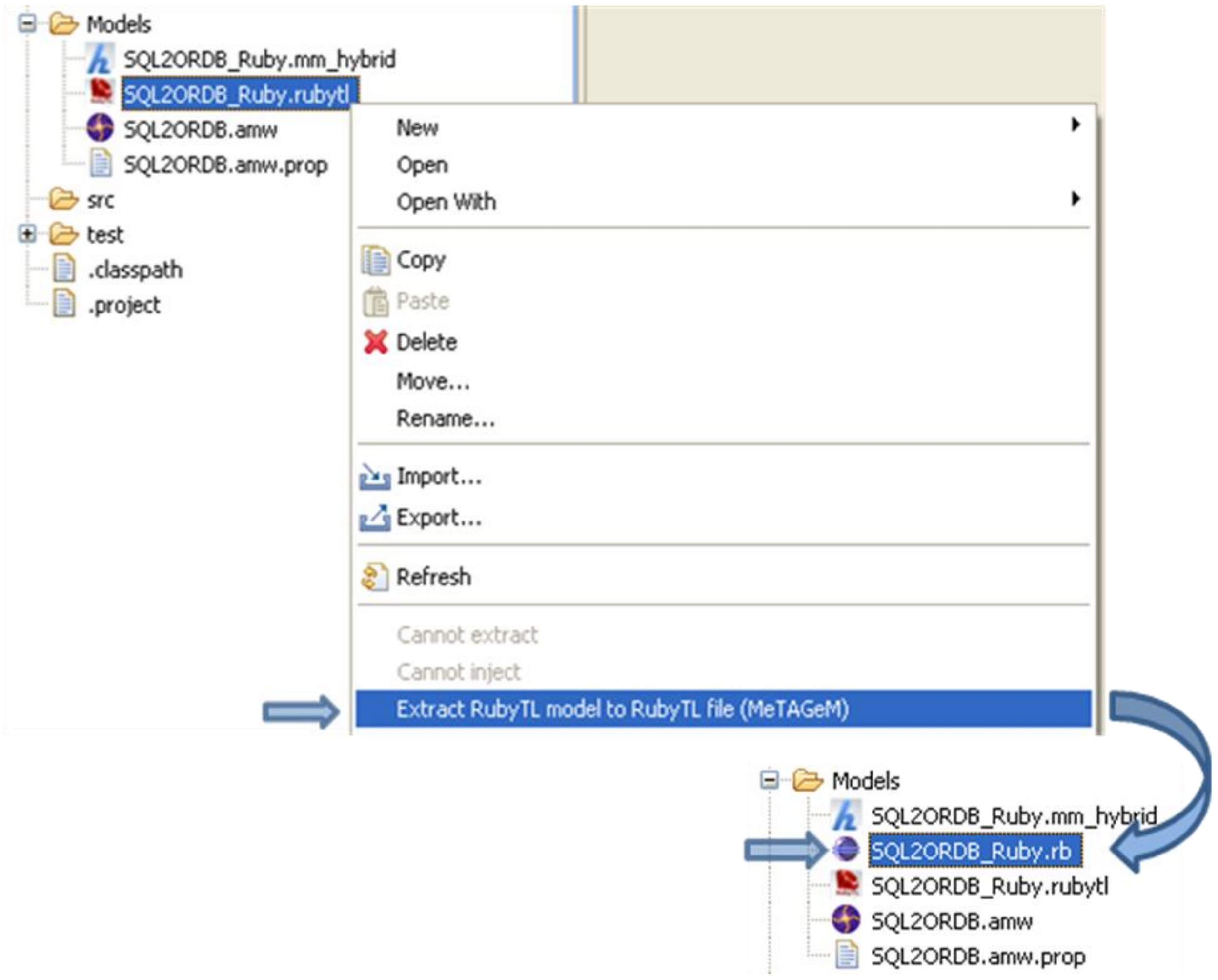


Fig. 24 – Generating RubyTL code from RubyTL model

Now, you have a RubyTL transformation file to be executed, however, as we have noticed in ATL case, in most cases, this generated code is not completely executable, so you must modify the code for converting it into a completely executable code.

We also notice that you must create a *rake* file **¡Error! No se encuentra el origen de la referencia.** to execute a RubyTL transformation.

7. References

[1] AMW – Atlas Model Weaver: http://www.eclipse.org/gmt/amw/

[2] ATL - Atlas Transformation Language: http://www.eclipse.org/atl/

[3] ATL User Manual – version 0.7, Febrero 2006

[4] RubyTL – Model Driven Development in Ruby: http://rubytl.rubyforge.org

[5] Transformation task from a Rake file: http://gts.inf.um.es/screencasts/introductory.html